BloRA MCMC: A Twist on Adaptive Metropolis-Hastings Algorithm

Ivy Liu, Zefan Liu

2024-04-19

1. Introduction

Markov Chain Monte Carlo (MCMC) methods are a class of algorithms for sampling from probability distributions based on constructing a Markov chain that has the desired distribution as its equilibrium distribution. MCMC is central to Bayesian statistics, allowing for the estimation of posterior distributions that are often analytically intractable. The basic idea behind MCMC is that by simulating the Markov chain over a long period, sampling from the Markov chain is equivalent to sampling from the target distribution.

Traditional MCMC methods include the Metropolis-Hastings (MH) algorithm and the Gibbs sampler. The effectiveness (the convergence rate) of traditional MCMC methods heavily depends on the choice of the proposal distribution and tuning parameters, such as the step size. These parameters need to be carefully selected and fixed throughout the sampling process. If poorly chosen, the chain might require a long time to converge to the target distribution, leading to inefficient sampling. A common approach to this problem is to run the MCMC for a certain period, tune the proposal distribution according to the acceptance rate, and use the tuned proposal to generate samples. This approach in reality, however, often yield undesirable results (Haario et al., 2001).

Alternatively, adaptive MCMC methods aim to overcome the limitations of traditional MCMC by allowing the algorithm to automatically adjust its parameters during the sampling process (Andrieu & Thoms, 2008). Specifically, the AM algorithm (Haario et al., 2001) uses a proposal distribution that is adjusted by the empirical estimate of the covariance structure of the target distribution based on the iterations so far. However, this algorithm may be limited because of the fixed MH proposal as part of its transition kernel. Meanwhile, it does not allow us to utilize the covariance of parameters flexibly. In this project, we aim to resolve the limitation by adapting the acceptance probabilities in a careful way. We propose a new scheme that adds more flexibility to the AM algorithm, such that its step size is also determined by the acceptance probability of the previous run. In addition, we allow such an adaptation to be applied to each parameter block separately.

In the next section, we give an introduction of our algorithm and a mathematical check on the π-invariance property of it. In Section 3, we test our algorithm on three simulated or real datasets, along with the comparison with the traditional MH algorithm. We conclude the report with a general discussion of our adaptive MCMC algorithm in Section 4, highlighting its distinctive features and potential applications.

2. Methodology

2.1 Algorithm

We propose the following MCMC algorithm, which imposes a twist on the proposal structure of Adaptive Monte Carlo (AMC) method (Andrieu & Thoms, 2008) and allows block-wise update of parameters. Specifically, for each block of parameters, our algorithm still uses the mixture of two multivariate normal proposals to update parameters. However, while we update the stepsize of the first proposal based on the empirical covariance matrix of this block (Andrieu & Thoms, 2008), we also update the stepsize of the second proposal based on the acceptance ratio of the previous iteration. Block can be chosen flexibly, so that different covariance structure can be used. Due to such a mechanism, we call our new algorithm **BloRA**, which stands for Blockwise Ratio-Adjusted AMC algorithm. The algorithm works as follows:

Algorithm 1 BloRA MCMC

Let D denote the number of parameter blocks to estimate, and $L = \{L_1, ..., L_D\}$ be a vector of dimension of each parameter block. Let the first subscript be the index of element, and the second subscript be the index of iteration. Initialize $\mathbf{\Gamma}_0 = \{\Gamma_{1,0}, ..., \Gamma_{D,0}\}, \mathbf{\Sigma}_0 = \{\Sigma_{1,0}, ..., \Sigma_{D,0}\}, \beta = 0.5$ and $\alpha = 0.4$. Let N denote the iteration times, x denote the list of samples for each parameter block, $\bar{x}$ denote the list of sample means, $\gamma(x)$ denote the unnormalized density of target distribution, and x_0 denote a list of initial values picked for each parameter block. Initialize $\bar{x}$ and x as x_0, and initialize $\mathbf{S}$ as a list of empty matrices to store MCMC samples.
for $n = 1, ..., N$ **do**
  $W_n \leftarrow \frac{\alpha_{n-1} - 0.4}{\sqrt{n}}$
  Initialize an empty list x'
  for parameter block $d = 1, ..., D$ **do**
    $\Gamma_{d,n} = (1 + W_n)\Gamma_{d,n-1}$
    $x'_d \sim Q_{d,n}(x_{d,n}|x_{d,n-1}) = (1 - \beta)N(x_{d,n-1}, (2.38)^2\Sigma_{d,n}/L_d) + \beta N(x_{d,n-1}, 0.1 \times \Gamma_{d,n}/L_d)$
  end for
  ratio $= \gamma(x')/\gamma(x_{n-1})$
  if runif(1) $<$ ratio **then**
    $x_n = x'$
  else $x_n = x_{n-1}$
  end if
  for parameter block $d = 1, ..., D$ **do**
    $\mathbf{S}_d[n,] = x_{d,n}$
  end for
  for parameter block $d = 1, ..., D$ **do**
    if $n = 2$ **then**
      $\Sigma_{d,n} =$**Cov**$(\mathbf{S}_d[1:2,])$
    else if $n > 2$ **then**
      $\Sigma_{d,n} = \frac{n-1}{n}\Sigma_{d,n-1} + \frac{1}{n^2}(\bar{x}_{d,n-1} - x_{d,n})(\bar{x}_{d,n-1} - x_{d,n})^T$
      $\bar{x}_{d,n} = ((n-1)\bar{x}_{d,n-1} + x_{d,n})/n$
    end if
  end for
end for
Return $\mathbf{S}$, a list of matrices of MCMC samples.

2.2 Proof of π-Invariance

Let $\mathcal{L}$ be the empirical density of the samples we generated from the **BloRA** sampler, π be the target distribution, and g be the test function. To prove that **BloRA** guarantees $\lim_{n\to\infty} \|\mathcal{L}(X_n) - \pi(\cdot)\| = 0$ (asymptotic convergence) and $\lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} g(X_i) = \pi(g)$ (WLLN), we need to prove our algorithm satisfy the following two conditions (Roberts & Rosenthal, 2007):

1. **Diminishing Adaptation**: $\lim_{n\to\infty}\sup_{x\in\mathcal{X}}||P_{\Gamma_{n+1}}(x) - P_{\Gamma_n}(x)||$ is $o_p(1)$ where $||\cdots||$ denotes the total variation distance.
2. **Bounded Convergence**: The time that distribution that our algorithm preserves converges to the target distribution is finite.

Lemma 1. ***BloRA*** *maintains diminishing adaptation.*

Proof. First consider only one block of parameters, and fix any $\mathbf{x} \in \mathcal{X}$ in that block. This is possible because we have independent updates among blocks. The transition kernel in our definition is

$$P_{\Gamma_n}(\mathbf{x}) = Q_n(\cdot|\mathbf{x}) := (1-\beta)\; Q_{n,1}(\cdot|\mathbf{x}) + \beta\; Q_{n,2}(\cdot|\mathbf{x})$$

where $Q_{n,1}(\cdot|\mathbf{x})$ is the empirical proposal and $Q_{n,2}(\cdot|\mathbf{x})$ is the adjusted random walk proposal. It remains to show that the deminishing property holds for both Q_1 and Q_2, and after that we can use apply the triangular inequality to the total variation distance, and show that the diminishing property holds for Q.

First focus on Q_2. Q_2 is a multivariate normal proposal centered at x. The total variation distance is related to the KL divergence by Pinsker's inequality,

$$||Q_{n+1,2}(\cdot|\mathbf{x}) - Q_{n,2}(\cdot|\mathbf{x})|| \leq \sqrt{\frac{1}{2} D_{KL}[Q_{n+1,2}(\cdot|\mathbf{x}) \;||\; Q_{n,2}(\cdot|\mathbf{x})]}$$

To simplify notation, we let $\Sigma_{n,2}$ to be the covariance matrix of $Q_{n,2}$.For two multivariate normal distributions, the KL divergence can be simplified to

$$\begin{aligned}
D_{KL}[Q_{n+1,2}(\cdot|\mathbf{x}) \;||\; Q_{n,2}(\cdot|\mathbf{x})] &= \mathbb{E}_{Q_{n+1,2}}[\log(Q_{n+1,2}(\cdot|\mathbf{x})) - log(Q_{n,2}(\cdot|\mathbf{x}))] \\
&= \mathbb{E}_{Q_{n+1,2}}\left[\frac{1}{2}\log\frac{|\Sigma_{n+1,2}|}{|\Sigma_{n,2}|} - \frac{1}{2}(\mathbf{y}-\mathbf{x})^T\Sigma_{n+1,2}^{-1}(\mathbf{y}-\mathbf{x}) + \frac{1}{2}(\mathbf{y}-\mathbf{x})^T\Sigma_{n,2}^{-1}(\mathbf{y}-\mathbf{x})\right] \\
&= \frac{1}{2}\mathbb{E}_p\left[\log\frac{|\Sigma_q|}{|\Sigma_p|}\right] - \frac{1}{2}\mathbb{E}_{Q_{n+1,2}}\left[(\mathbf{y}-\mathbf{x})^T\Sigma_{n+1,2}^{-1}(\mathbf{y}-\mathbf{x})\right] + \frac{1}{2}\mathbb{E}_{Q_{n+1,2}}\left[(\mathbf{y}-\mathbf{x})^T\Sigma_{n,2}^{-1}(\mathbf{y}-\mathbf{x})\right] \\
&= \ldots \text{ (some algebraic simplification)} \\
&= \frac{1}{2}\left(\log\frac{|\Sigma_{n+1,2}|}{|\Sigma_{n,2}|} - k + (\mathbf{x}-\mathbf{x})^T\Sigma_{n,2}^{-1}(\mathbf{x}-\mathbf{x}) + tr\left\{\Sigma_{n,2}^{-1}\Sigma_{n+1,2}\right\}\right)
\end{aligned}$$

Since $\Sigma_{n+1,2} = (1+W_{n+1})\Sigma_{n,2}$ by our construction, and $W_{n+1} = \frac{\alpha_n - 0.4}{\sqrt{n}} \to 0$ as $n \to \infty$, the first term in the last equation shrinks to zero. For the third term, since we center $Q_{n+1,2}$ and $Q_{n,2}$ at the same point, it

also shrinks to zero. As for the last term, using the same trick $\Sigma_{n+1,2} = (1 + W_{n+1})\Sigma_{n,2}$ again, we find that $tr\left\{\Sigma_{n,2}^{-1}\Sigma_{n+1,2}\right\} = \frac{1}{1+W_n}k \to k$ as $n \to \infty$. Therefore, the KL divergence $D_{KL}[Q_{n+1,2}(\cdot|\mathbf{x}) \,||\, Q_{n,2}(\cdot|\mathbf{x})]$ goes to zero as n goes to infinity, and thus the total variation distance, i.e.

$$\lim_{n\to\infty} ||Q_{n+1,2}(\cdot|\mathbf{x}) - Q_{n,2}(\cdot|\mathbf{x})|| = 0 \text{ in probability}$$

Since we begin with an arbitrary $\mathbf{x}$, taking the sup does not violate the above equality. Hence, the diminishing adaptation holds for Q_2.

Repeat the above derivation for Q_1, we get exact the same divergence for Q_1, i.e.

$$D_{KL}[Q_{n+1,1}(\cdot|\mathbf{x}) \,||\, Q_{n,1}(\cdot|\mathbf{x})] = \frac{1}{2}\left(\log\frac{|\Sigma_{n+1,1}|}{|\Sigma_{n,1}|} - k + (\mathbf{x}-\mathbf{x})^T\Sigma_{n,1}^{-1}(\mathbf{x}-\mathbf{x}) + tr\left\{\Sigma_{n,1}^{-1}\Sigma_{n+1,1}\right\}\right)$$

Using the Sherman-Morrison formula,

$$\Sigma_{n+1,1} = \frac{n}{n+1}\Sigma_{n,1} + \frac{n-1}{n^2}(\bar{\mathbf{x}}_\mathbf{n} - \mathbf{y})(\bar{\mathbf{x}}_\mathbf{n} - \mathbf{y})^T$$

Plugging in to the KL divergence expression, we can easily yield

$$\lim_{n\to\infty} \log\frac{|\Sigma_{n+1,1}|}{|\Sigma_{n,1}|} = 0$$

and

$$\lim_{n\to\infty} tr\left\{\Sigma_{n,1}^{-1}\Sigma_{n+1,1}\right\} = k,$$

which implies

$$\lim_{n\to\infty} \sup_{\mathbf{x}\in\mathcal{X}} ||Q_{n+1,1}(\cdot|\mathbf{x}) - Q_{n,1}(\cdot|\mathbf{x})|| = 0 \text{ in probability}$$

Now we have shown that the kernel $Q_{d,n}$ has diminishing adaptation property. The overall kernel Q_n can be considered as a mixture of $\{Q_{d,n}\}_{d=1,\cdots,D}$. The mixture of π-invariant kernels is also π-invariant, which completes the proof.

□

The π-invariance of the mixture kernel is proved in the course page (Bouchard-Côté, 2024).

The bounded convergence condition is harder to prove. However, previous work has shown that an adaptive MCMC is ergodic if it has diminishing adaptation and satisfies other technical conditions that are easier to check (Bai et al., 2011). Following Theorem 6 in this paper, we conclude that

Theorem 2. ***BloRA*** *is ergodic, and thus π-invariant if the following conditions are satisfied:*

1. *Lighter-than-exponential tail of the target distribution: The target density π is positive and has continuous first derivative, such that*
$$\limsup_{|x|\to\infty}\langle\frac{x}{|x|}, \nabla\log\pi(x)\rangle = -\infty$$
2. *Regularity condition on the target distribution: π is absolutely continuous with respect to Lebesgue measure μ_d, with π bounded away from zero and infinity on compact sets, and $\sup_{x\in\mathcal{X}}\pi(x) < \infty$*

3. Strongly decreasing condition on the target distribution: π has continuous first derivatives such that

$$-\limsup_{|x|\to\infty}\langle\frac{x}{|x|},\frac{\nabla\pi(x)}{|\nabla\pi(x)|}\rangle>0$$

The three conditions on the target distribution are not too harsh. Many common distributions in the exponential family that has full support satisfy these conditions. In the following, we provide a proof sketch of the theorem.

Proof. Following Theorem 6, it suffices to show that the proposal of **BloRA** is uniformly locally positive given the above conditions hold. Mathematically, we would like to show that

$$\exists\zeta>0, \text{ such that } \omega:=\inf_{\gamma\in\mathcal{Y}}\inf_{|z|\in\zeta}q_\gamma(z)>0,$$

where $\{q_\gamma:\gamma\in\mathcal{Y}\}$ is the family of transition kernels we have.

Since our kernel is a mixture, i.e. a weighted sum, of two normal distribution kernel centered at some $x\in\mathcal{X}$, the above statement is obvious if we can ensure that one of the covariance matrices of the two normal proposals is strictly positive.

For the random walk proposal, we initialize its covariance matrix to be some diagonal matrix. Within finite iterations, there is a lower bound on each entry of the diagonal matrix. Hence, within the neighborhood $|z|\in\zeta$, $q_\gamma(z)>0$ for any γ.

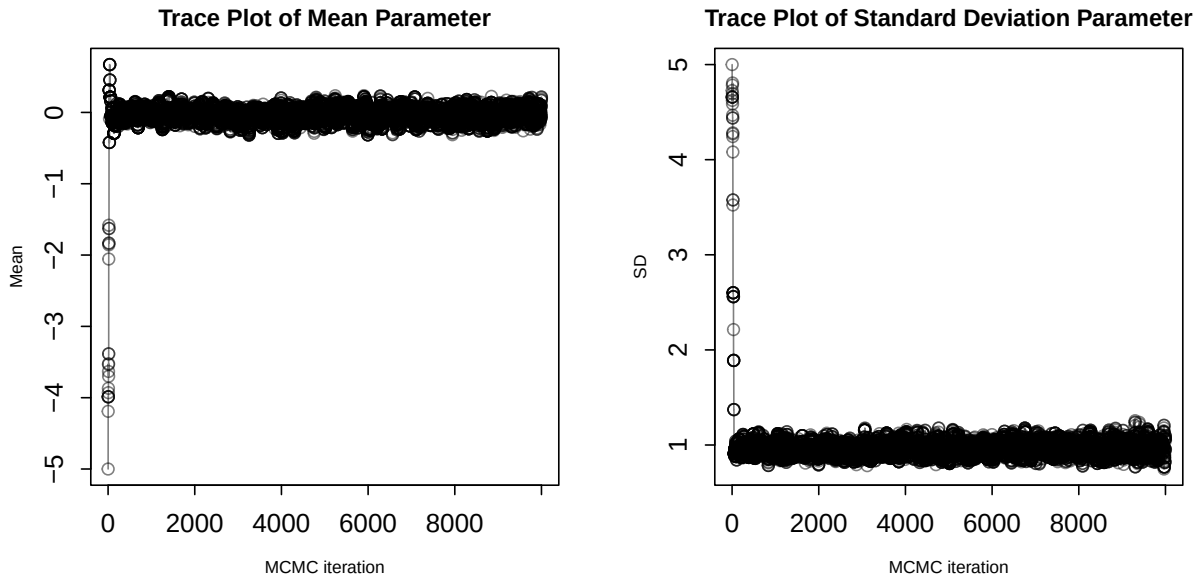
□

Notice that we can also introduce a hard lower bound on the covariance matrix of the random walk proposal. For example, for each diagonal entry of Γ_n, $\Gamma_n^{d,d}$, we can let $\Gamma_n^{d,d}=\max\{(1+W_n)\Gamma_{n-1}^{d,d},0.1\}$. This slight modification makes the above proof robust on infinite runs.

3. Simulation

3.1 Simulation under simple normal case

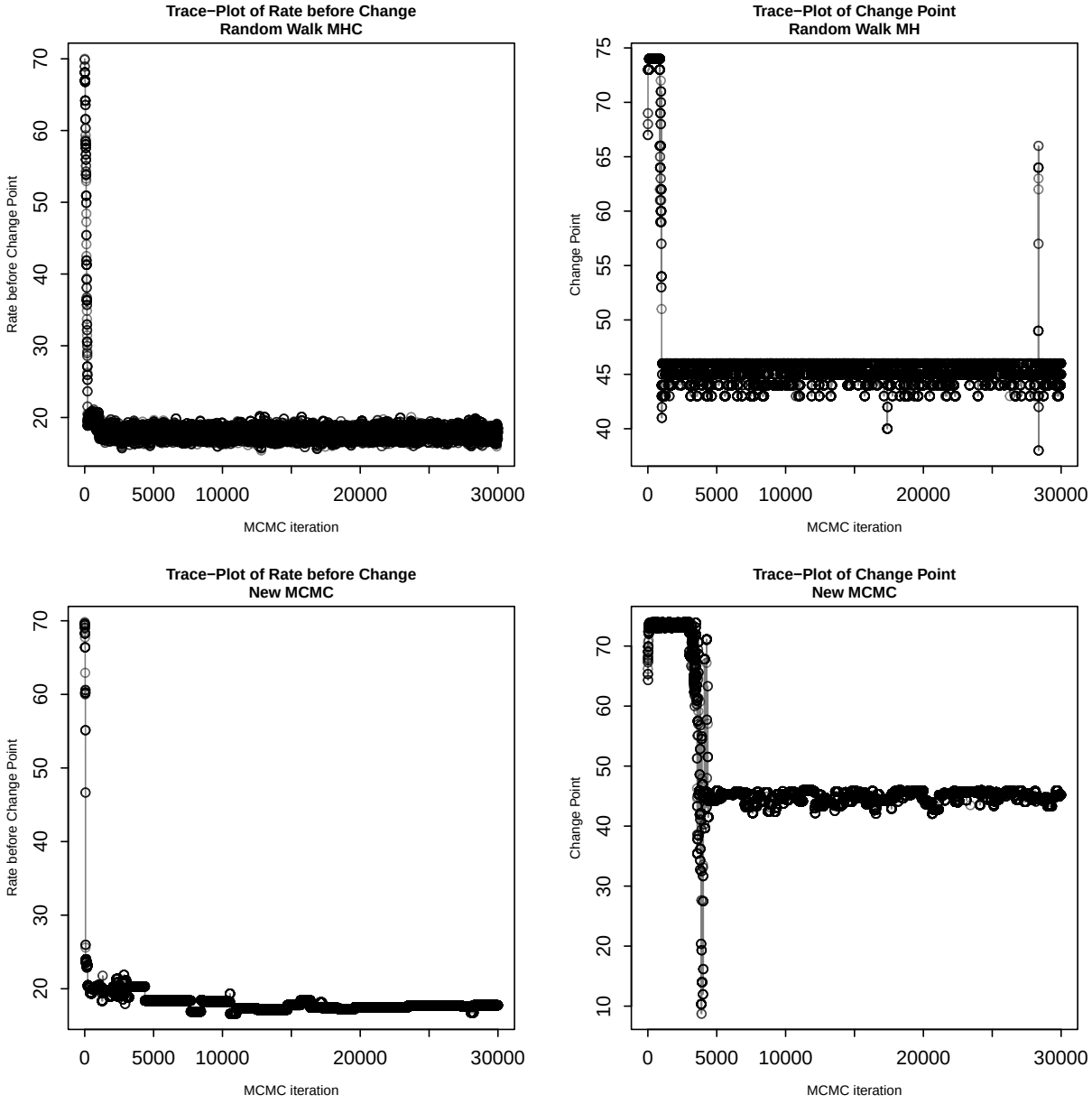
Our simulation begins with a toy example which employs a standard normal distribution. Specifically, the algorithm generates 100 independent and identically distributed (i.i.d.) data points, each drawn from a standard normal distribution. We use **BloRA** sampler to estimate the mean and standard deviation parameters of the distribution. This simulation provides a structured environment to evaluate the efficacy of **BloRA** algorithm. Initial values for the mean and standard deviation are set at -5 and 5 respectively. Over the course of 10,000 iterations, the trace plots for these two parameters are as follows:



In the first plot, following rapid mixing, the trace of the mean parameter appears to oscillate around 0, which aligns with the true mean of a standard normal distribution. The dispersion of points around true value indicates the variability in the mean estimates, which is due to the small sample size. In addition, the absence of trend and heteroscedasticity suggest that the Markov chain is stationary. The observations on the second plot lead to similar conclusions, which confirm that our proposed MCMC algorithm has performed effectively in estimating these parameters. We consider this toy example as a model diagnosis.

3.2 Simulation using in-class dataset: SMS data

The second simulation study uses a real-world dataset under a mixture model assumption, which is directly borrowed from course materials. The dataset records the daily number of text messages sent and received by Davidson Pilon over 74 days, and it's assumed that there exist a "hidden" change point among these 74 days, such that the data before and after the change point come from two distinct exponential distributions. Our MCMC algorithm aims to identify the location of this change point and estimate the parameters of two exponential distributions under a Bayesian model framework. The initial value is intentionally set to be extreme to test the robustness of our algorithm, and a comparative analysis is made between our algorithm and the random work mixture MH that we examined in exercise 9. We encapsulate the two exponential parameters within block 1 and position the change point in block 2. The trace plots illustrating the results are presented below:



The comparison between the trace-plots of the same parameters suggests that our algorithm can still mix rapidly and provide robust estimations under extreme initial conditions, using the results from traditional MH as a benchmark. To further explore which block structure optimizes **BloRA**'s performance in this scenario, we test three variants of **BloRA** relative to traditional MH. Specifically, one employing empirical covariance across all three parameters, another using empirical covariance between the two exponential parameters, and a third variant without any empirical covariance. To mitigate the effect of the burn-in stage on our computations, we consider only the samples after 10,000 iterations. Trace-plots (for the change point only) and a summary table of their effective sample size per second (ESS/second) are presented below:

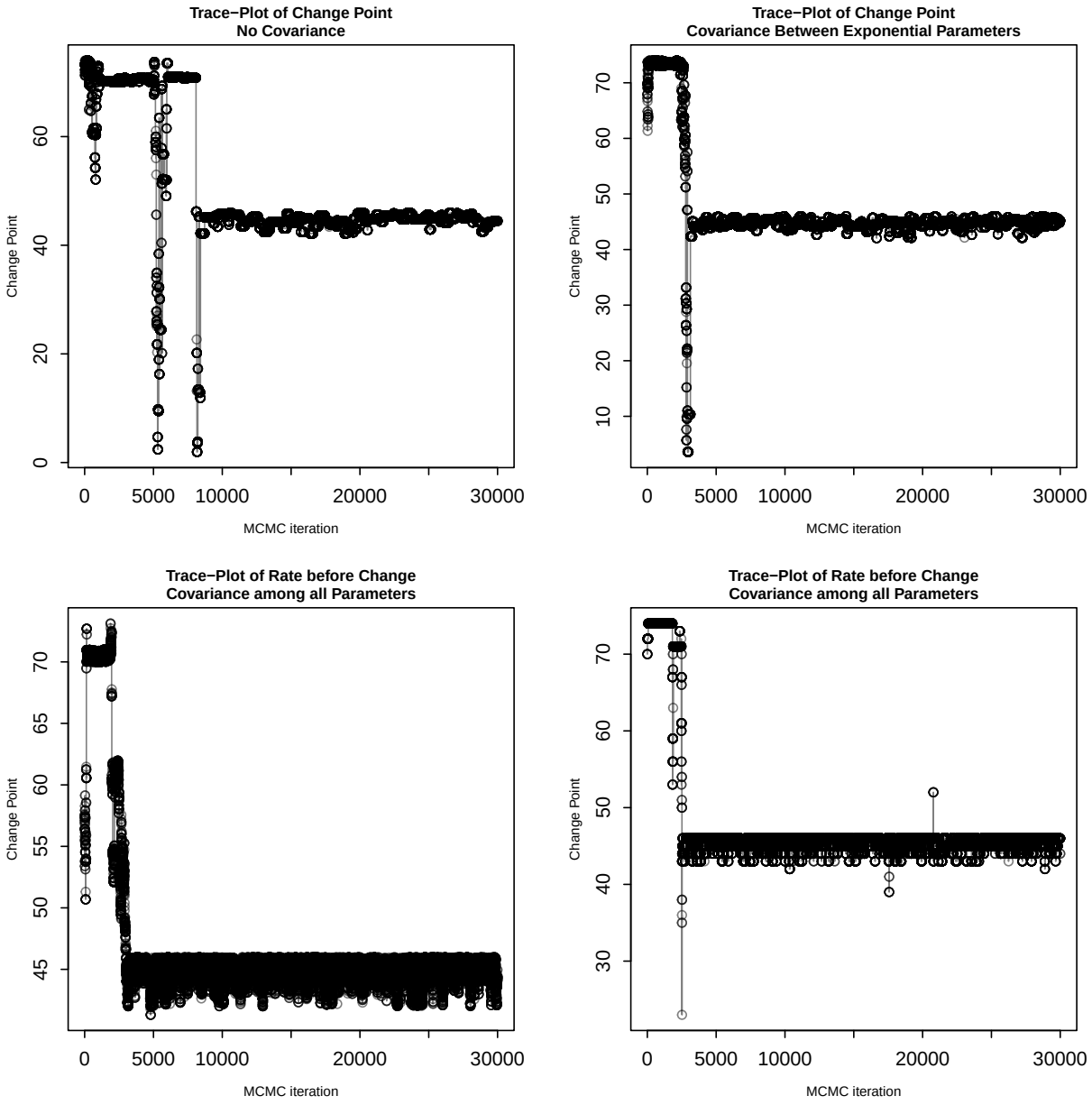


Table 1: The Summary of ESS/second of three candidates from BloRA

	No Covariance	Covariance between Two Exponential Parameterss	Covariance among all Parameters	Traditional MH
Exponential Rate before Change Point	28.553887	20.47606	19.46866	381.54432
Exponential Rate after Change Point	20.247492	29.17646	15.04260	225.87039
Change Point	5.082036	15.04604	92.56443	99.74353

4. Conclusion

In this project, we introduced a novel adaptive MCMC algorithm, **BloRA**, which enhances flexibility by incorporating blockwise parameter design and ratio-adjusted step sizes. As demonstrated in Section 2, **BloRA** maintains ergodicity and π-invariance. Furthermore, we conducted two simulation studies to illustrate that this new algorithm offers a promising result. In particular, it can automatically tune the proposal step size, which is theoretically more robust if we have no prior information on the spread of the parameters of interest. However, it is important to acknowledge some potential limitations. The performance of the algorithm is significantly influenced by its blockwise design, yet finding the optimal blocks remains a challenge that often depends heavily on prior knowledge. Overall, **BloRA** demonstrates superior performance in scenarios involving complex target distributions and correlated parameters.

References

Andrieu, C., & Thoms, J. (2008). A tutorial on adaptive MCMC. *Statistics and Computing*, *18, issue 4*, 343–373. https://doi.org/10.1007/s11222-008-9110-y

Bai, Y., Roberts, G., & Rosenthal, J. (2011). On the containment condition for adaptive markov chain monte carlo algorithms. *Adv. Appl. Stat.*, *21*.

Bouchard-Côté, A. (2024). *STAT447C: Bayesian statistics*. https://ubc-stat-ml.github.io/web447/

Haario, H., Saksman, E., & Tamminen, J. (2001). An adaptive Metropolis algorithm. *Bernoulli*, *7*(2), 223–242.

Roberts, G. O., & Rosenthal, J. S. (2007). Coupling and ergodicity of adaptive markov chain monte carlo algorithms. *Journal of Applied Probability*, *44*(2), 458–475. http://www.jstor.org/stable/27595854

5. Appendix

5.1 MCMC Implementation

Following is the detailed implementation of **BloRA**. The `kernel()` function gives **BloRA** proposal state, the `mcmc()` is a single-block version, and the `mcmc_multmix()` is the **BloRA** demonstrated in Algorithm 1.

```
# run mcmc once
kernel <- function(Gamma, x, x_sigma, beta, d){
  kernel_choice <- rbinom(1,1,prob=beta)
  if(kernel_choice==0){
    z <- MASS::mvrnorm(n = 1, mu=rep(0,d), Sigma = (2.38)^2 * x_sigma/d)
  }
  else{
    z <- MASS::mvrnorm(n = 1, mu=rep(0,d), Sigma = 0.1 * Gamma/d)
  }
  x_prime = x + z
  # return the next state
  return(x_prime)
}
```

```
## x0: initial state
## N: number of iterations
## log_gamma: unnormalized posterior distribution on log scale
## Gamma: covariance of a random walk step
## beta: weight for the random walk in the mixture proposal
## alpha: initial Hastings ratio
## x_sigma: empirical covariance matrix
## dat: the observed data
mcmc <- function(x0, N, log_gamma, Gamma, beta, alpha, x_sigma, dat){
  d <- length(x0) # dim of x0
  samples <- matrix(0, nrow = N, ncol = d)
  xbar <- x0 # current xbar
  x = x0

  # start the iteration
  for(t in 1:N){
    Wn <- (alpha - 0.4)/sqrt(t)
    Gamma <- (1 + Wn) * Gamma
    x_prime <- forward_once(Gamma, x, x_sigma, beta, d)
    alpha <- min(c(1, exp(log_gamma(x_prime, dat) - log_gamma(x, dat))))

    #determine whether to update
    if(runif(1) < alpha){
```

```
      x = x_prime
    }

    # update covariance matrix using Sherman-Morrison formula
    if(t == 2){
      x_sigma = var(samples[1:2, ])
    } else if(t > 2){
      x_sigma <- (t-1)/t * x_sigma + (t-1)/t^2 * (xbar - x) %*% t(xbar - x)
    }
    # update xbar
    xbar = ((t-1)*xbar + x)/t

    # store new sample
    samples[t,] <- x
  }
  return(samples)
}
```

```
mcmc_multmix <- function(x0, N, log_gamma, Gamma, beta, alpha, x_sigma, dat){
  d_list <- sapply(x0, length) # dim of each parameter
  d <- sum(d_list)
  xbar <- x0
  x <- x0
  samples <- lapply(d_list, function(d){matrix(data=0, nrow=N, ncol=d)})

  # start the iteration
  for(t in 1:N){
    Wn <- (alpha - 0.4)/sqrt(t)
    Gamma <- lapply(Gamma, function(gamm){(1 + Wn) * gamm})
    x_prime <- list()

    # get new samples
    for(i in 1:length(d_list)){
      x_prime[[i]] <- kernel(Gamma[[i]], x[[i]], x_sigma[[i]], beta, d_list[i])
    }

    #MH accept/reject
    alpha <- min(c(1, exp(log_gamma(unlist(x_prime),dat)-
                          log_gamma(unlist(x), dat))))
    if(runif(1) < alpha){
      x = x_prime
    }
```

```
    # store new sample
    for(i in 1:length(d_list)){
      samples[[i]][t,] <- x[[i]]
    }

    # update covariance matrix using Sherman-Morrison formula
    if(t == 2){
      x_sigma = lapply(samples, function(sample){var(sample[1:2, ])})
    } else if(t > 2){
      x_sigma <- mapply(function(sigma, mu, y) {
        (t-1)/t * sigma + (t-1)/t^2 * (mu - y) %*% t(mu - y)
        }, x_sigma, xbar, x, SIMPLIFY = FALSE)
    }

    # update xbar
    xbar = mapply(function(mu, y) {((t-1)*mu + y)/t},
                  xbar, x, SIMPLIFY = FALSE)
  }

  return(samples)
}
```

5.2 Misc

The code snippet below presents an alternative approach we experimented with. This twisted version proposes new states that are generally inversely correlated with the current state, which should theoretically enhance the mixing rate of the Markov chain. However, the unique characteristics of the SMS dataset led to inconsistent performance with this sampler. So we precluded this version in our main report.

```
mcmc_multmix_reverse <- function(x0, N, log_gamma, Gamma, beta, alpha, x_sigma, dat){
  d_list <- sapply(x0, length) # dim of each parameter
  d <- sum(d_list)
  xbar <- x0
  xbar_short <- x0
  x <- x0
  samples <- lapply(d_list, function(d){matrix(data=0, nrow=N, ncol=d)})

  # create the function reverse-kernel
  reverse_kernel <- function(Gamma, x, x_sigma, xbar_short, beta, d){
  kernel_choice <- rbinom(1,1,prob=beta)
  if(kernel_choice==0){
    x_prime <- MASS::mvrnorm(n = 1, mu=2*xbar_short-x,
                             Sigma = (2.38)^2 * x_sigma/d)
```

```
  }
  else{
    x_prime <- MASS::mvrnorm(n = 1, mu=2*xbar_short-x, Sigma = 0.1 * Gamma/d)
  }
  # return the next state
  return(x_prime)
}

  # start the iteration
  for(t in 1:N){
    Wn <- (alpha - 0.4)/sqrt(t)
    Gamma <- lapply(Gamma, function(gamm){(1 + Wn) * gamm})
    x_prime <- list()

    # get new samples
    for(i in 1:length(d_list)){
      x_prime[[i]] <- reverse_kernel(Gamma[[i]], x[[i]], x_sigma[[i]],
                                     xbar_short[[i]],beta, d_list[i])
    }

    #MH accept/reject
    alpha <- min(c(1, exp(log_gamma(unlist(x_prime),dat)-
                            log_gamma(unlist(x), dat))))
    if(runif(1) < alpha){
      x = x_prime
    }

    # store new sample
    for(i in 1:length(d_list)){
      samples[[i]][t,] <- x[[i]]
    }

    # update covariance matrix using Sherman-Morrison formula
    if(t == 2){
      x_sigma = lapply(samples, function(sample){var(sample[1:2, ])})
    } else if(t > 2){
      x_sigma <- mapply(function(sigma, mu, y) {
        (t-1)/t * sigma + (t-1)/t^2 * (mu - y) %*% t(mu - y)
        }, x_sigma, xbar, x, SIMPLIFY = FALSE)
    }

    # update xbar
    xbar = mapply(function(mu, y) {((t-1)*mu + y)/t},
```

```
                  xbar, x, SIMPLIFY = FALSE)
    if(t<=2000){
      xbar_short = xbar
    }
    else{
      xbar_short = mapply(function(mu_short, y, sam){
        (mu_short*2000-sam[t-2000,]+y)/2000
      }, xbar_short, x, samples, SIMPLIFY = FALSE)
    }
  }

  return(samples)
}
```